Why MOAB's iMesh Implementation Does Not Support All Iterator Semantics for List-Type Sets

The iMesh 1.2 specification requires that iterators over list-type entity sets be updated in response to membership changes in the set. Specifically, if entities are added to or removed from the set, the spec requires that added entities be iterated without needing to reset the iterator, and that removed entities not be iterated. MOAB does not support this capability, with the following rationale:

- MOAB's iMesh iterators are implemented on top of the MOAB API, rather than being native to MOAB proper; updating all existing iMesh iterators on changes to set membership would therefore be restricted to set membership changes requested through the iMesh API only. However, an important use case for iMesh is where an application interacts with MOAB's API, then calls an iMesh-based service that operates on MOAB data. Iterator updates implemented in the iMesh layer would miss any set updates made through the MOAB API.

- MOAB uses array-based storage for its list-type sets. The actual data container used internally varies, depending on how many entities are in the set. A MOAB-native iterator would have to account for set membership changes, as well as the various types of storage used. This code would be difficult to maintain, given the number of storage types and the various types of modifications that would have to be supported.

- The iMesh spec requires iterators to account for entity removals from sets. Properly updating an iterator in this case would require searching the iterator contents before or after the current iterator position (it would not be enough to search the set contents; the contents of the iterator depend on the type and topology of the iterator). This would impose even more cost on an already-expensive operation.

- The intent of this requirement is to simplify coding for mesh modification algorithms. We argue that skipping entities removed from a set, as well as iterating over newly-added entities, could be easily accomplished by marking the former and keeping a separate list of the latter.